

Threat Hunt Notes

Fundamental Information

Hunt Name: Deep Access: The Adversary (CTF)

Initiated By: Victor Cardoso

Start Date and Time: Jun 10, 2025 12:10 UTC

End Date and Time: Jun 14, 2025 06:21 UTC

Affected Asset Information

Name: acolyte756

OS: Windows 10 22H2

Private IP: 10.0.0.130

Main Account: acolyte756\acolaight

First seen: May 25, 2025 6:35:57 UTC

Last seen: May 25, 2025 11:38:08 UTC

Tools and Technologies Used

- Azure Log Analytics Workspace
- [Row Zero](#)
- Microsoft Endpoint Defender
- KQL

Notes and Findings

Starting Point/Flag 0

Before you officially begin the flags, you must first determine where to start hunting. The attack points are randomized, but you may want to start with the newly created virtual machines that were only active for a few hours before being deleted, implying that the device(s) did not generate thousands of recorded processes, at least not in the central logging repository.

a. Around May 24th 2025

What could be observed in the “Starting Point” was the hints dropped to find the specific machine, these be:

- Newly created virtual machines that were only active for a few hours before being deleted.
- Device did not generate thousands of recorded processes.
- May 24th 2025

First I came up with this KQL that traces all the machines created in the span of 24H in the mentioned date.

```
let start = datetime(2025-05-24);
let end = datetime(2025-05-25);
AzureActivity
| extend VMName = tostring(parse_json(Properties).resource)
| where TimeGenerated between (start .. end)
| where OperationNameValue in ("MICROSOFT.COMPUTE/VIRTUALMACHINES/WRITE",
"MICROSOFT.COMPUTE/VIRTUALMACHINES/DELETE")
| project TimeGenerated, OperationNameValue, ResourceGroup, Resource, Caller, VMName
| order by TimeGenerated asc
```

But after receiving about 6258 different results, further filtering was necessary.

After debating with Chat GPT (about his poor KQL skills) we came up with this query that **narrow it down** to isolate the **short-lived VMs**

```
let start = datetime(2025-05-24);
let end = datetime(2025-05-25);
let VMHUNT = AzureActivity
| where TimeGenerated between (start .. end)
| where OperationNameValue in ("MICROSOFT.COMPUTE/VIRTUALMACHINES/WRITE",
"MICROSOFT.COMPUTE/VIRTUALMACHINES/DELETE")
| extend VMName = tostring(parse_json(Properties).resource)
| summarize
    CreateTime = minif(TimeGenerated, OperationNameValue ==
"MICROSOFT.COMPUTE/VIRTUALMACHINES/WRITE"),
    DeleteTime = minif(TimeGenerated, OperationNameValue ==
"MICROSOFT.COMPUTE/VIRTUALMACHINES/DELETE")
    by ResourceGroup, VMName
| extend LifetimeMinutes = datetime_diff("minute", DeleteTime, CreateTime)
| where isnotempty(DeleteTime) and LifetimeMinutes < 240 // under 4 hours
| sort by CreateTime asc;
VMHUNT
```

I STILL FAILED!!!

So I decided to think more and have a really good look at the system.

First, I discovered that ActivitySubstatusValue has the value **“Created”** when virtual machines are created.

I also thought that since the CTF virtual machines were created by the same group of people (**BTW shout out to Josh and your team if you are reading this, and also btw, sorry**), they must have the same resources.

So when I went to the virtual machine created in the previous CTF, **“anthony-001”**, the resource group **“STUDENT-RG-9F70B6B2EAD907B656636D76BA0E504891F1D33097BA8D30CF1F955AB91F00D3”** was found, and after checking the logs of this Resource Group, BINGO!!! The machine named **“acolyte756”** was successfully found.

```
let start = datetime(2025-05-24);
let end = datetime(2025-05-25);
AzureActivity
| where TimeGenerated between (start .. end)
| where OperationNameValue == "MICROSOFT.COMPUTE/VIRTUALMACHINES/WRITE"
| where ResourceGroup ==
"STUDENT-RG-9F70B6B2EAD907B656636D76BA0E504891F1D33097BA8D30CF1F955AB91F00D3"
| extend VMName = Properties_d.resource
```

I know this literally sounds like cheating, it is. But hey... we're in the real world, in real world situations, so let's call it **“thinking outside the box”**. I thought like an adversary, or in this case, like the CTF designers/deployers.

TimeGenerated [UTC]	VMName	OperationNameValue	Level	ActivityStatusValue
> 5/24/2025, 8:15:48.620 PM	acolyte756	MICROSOFT.COMPUTE/VIRTUA...	Information	Success
> 5/24/2025, 8:05:45.951 PM	acolyte756	MICROSOFT.COMPUTE/VIRTUA...	Information	Accept
> 5/24/2025, 8:05:45.295 PM	acolyte756	MICROSOFT.COMPUTE/VIRTUA...	Information	Start
> 5/24/2025, 8:02:36.998 PM	acolyte756	MICROSOFT.COMPUTE/VIRTUA...	Information	Accept
> 5/24/2025, 8:02:36.513 PM	acolyte756	MICROSOFT.COMPUTE/VIRTUA...	Information	Start
> 5/24/2025, 4:04:40.080 PM	acolyte756	MICROSOFT.COMPUTE/VIRTUA...	Information	Accept
> 5/24/2025, 4:04:39.799 PM	acolyte756	MICROSOFT.COMPUTE/VIRTUA...	Information	Start
> 5/24/2025, 4:02:04.742 PM	acolyte756	MICROSOFT.COMPUTE/VIRTUA...	Information	Accept

Flag 1 – Initial PowerShell Execution Detection

Objective:

Pinpoint the earliest suspicious PowerShell activity that marks the intruder's possible entry.

What to Hunt:

Initial signs of PowerShell being used in a way that deviates from baseline usage.

Thought:

Understanding where it all began helps chart every move that follows. Look for PowerShell actions that started the chain.

After reviewing the requirements, I queried for everything between May 24 and May 25 and filter for processes started by either powershell or explorer.

Because of that I found the suspicious powershell process “**powershell.exe -Version 5.1 -s -NoLogo -NoProfile**” indicating an attacker trying to reduce footprint.

```
let start = datetime(2025-05-24);
let end = datetime(2025-05-26);
DeviceProcessEvents
| where DeviceName == "acolyte756"
| where Timestamp between (start .. end)
| where FileName contains "powershell" or InitiatingProcessFileName contains "powershell" or
InitiatingProcessFileName contains "explorer"
| project Timestamp, DeviceName, FileName, FolderPath, InitiatingProcessFileName,
InitiatingProcessCommandLine
| order by Timestamp asc
```

FLAG: 2025-05-25T09:14:02.3908261Z

May 25, 2025 10:...	acolyte756	conhost.exe	C:\Windows\System32\c... powershell.exe
Timestamp	May 25, 2025 10:14:02 AM		
DeviceName	acolyte756		
FileName	conhost.exe		
FolderPath	C:\Windows\System32\conhost.exe		
InitiatingProcessFileName	powershell.exe		
InitiatingProcessComma...	"powershell.exe" -Version 5.1 -s -NoLogo -NoProfile		

Flag 2 – Suspicious Outbound Signal

Objective:

Confirm an unusual outbound communication attempt from a potentially compromised host.

What to Hunt:

External destinations unrelated to normal business operations.

Thought:

When machines talk out of turn, it could be a sign of control being handed off.

Hint:

1. We don't have a controlled remote server
2. hollow tube

After correlating the timestamp with the Network Connections, it was found the suspicious remote server:
FLAG: **eoqsu1hq6e9ulga.m.pipedream.net**

DeviceNetworkEvents

```
| where DeviceName == "acolyte756"
| where Timestamp >= datetime(2025-05-25T09:14:02.3941533Z)
| order by Timestamp asc
```

Timestamp	DeviceId	DeviceName	ActionType	RemoteIP	RemotePort	RemoteUrl
> May 25, 2025 10:...	556d342e2d6fd2fbb...	acolyte756	ConnectionSuccess	(ip) 54.165.125.87	443	eoqsu1hq6e9ulga.m.pipedream.net

Flag 3 – Registry-based Autorun Setup

Objective:

Detect whether the adversary used registry-based mechanisms to gain persistence.

What to Hunt:

Name of the program tied to the new registry value created.

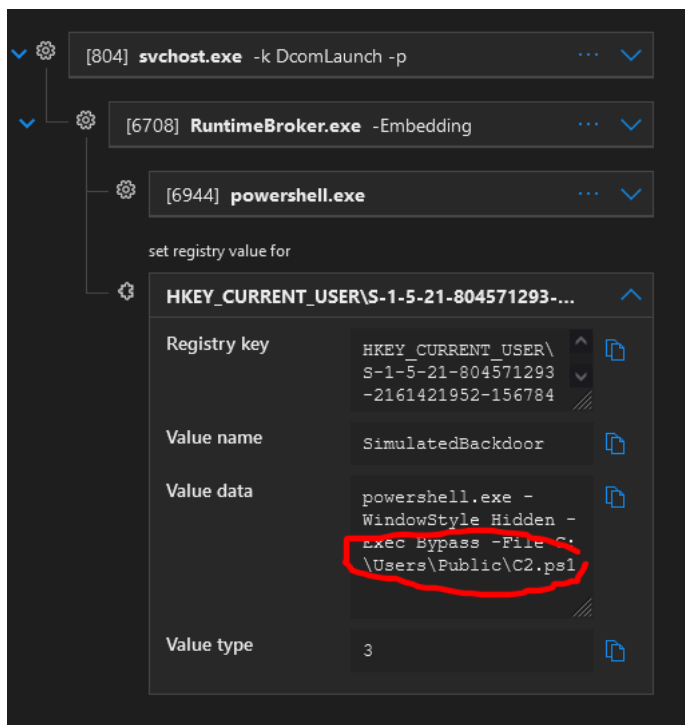
Thought:

Registry is a favored place to hide re-execution logic — it's reliable, stealthy, and usually overlooked. Identify the file associated with the newly created registry value *

After correlating the timestamp with the DeviceRegistryEvents, it was found the suspicious Registry
SimulatedBackdoor, and it relates to the file C:\Users\Public\C2.ps1
FLAG: **C2.ps1**

DeviceRegistryEvents

```
| where DeviceName == "acolyte756"  
| where Timestamp >= datetime(2025-05-25T09:14:02.3941533Z)  
| order by Timestamp asc
```



Flag 4 – Scheduled Task Persistence

Objective:

Investigate the presence of alternate autorun methods used by the intruder.

What to Hunt:

Verify if schedule task creation occurred in the system.

Thought:

Adversaries rarely rely on just one persistence method. Scheduled tasks offer stealth and reliability — track anomalies in their creation times and descriptions.

Again, I used `datetime(2025-05-25T09:14:02.3941533Z)` as a baseline for the search and found the Scheduled Task:

FLAG: **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\SimC2Task**

DeviceEvents

```
| where DeviceName == "acolyte756"  
| where ActionType == "ScheduledTaskCreated"  
| where Timestamp >= datetime(2025-05-25T09:14:02.3941533Z)  
| order by Timestamp asc
```

May 25, 2025 10:...	556d342e2d6fd2fbb...	acolyte756	Scheduled
Timestamp	May 25, 2025 10:14:05 AM		
DeviceId	556d342e2d6fd2fbb822230632014def1e6d58c2		
DeviceName	acolyte756		
ActionType	ScheduledTaskCreated		
InitiatingProcessAccount...	acolyte756		
InitiatingProcessAccount...	acolaight		
InitiatingProcessAccount...	S-1-5-21-804571293-2161421952-1567847933-500		
InitiatingProcessLogonId	952785		
ReportId	2770		
AdditionalFields	{ "TaskName": "\\SimC2Task", "TaskContent": "{ \"Triggers\": { \"Logon		
TaskName	\\SimC2Task		
TaskContent	{ \"Triggers\": { \"LogonTrigger\": { \"Enabled\": \"true\" }, \"Actions\": { \"Exec		

Flag 5 – Obfuscated PowerShell Execution
Objective:
Uncover signs of script concealment or encoding in command-line activity.
What to Hunt:
Look for PowerShell patterns that don't immediately reveal their purpose — decoding may be required.
Thought:
Encoding is a cloak. Finding it means someone may be hiding something deeper within an otherwise familiar tool.
Hint:
1. "Simulated obfuscated execution"

After using the timestamp as baseline and reviewing the DeviceRegistryEvents with less filters, the process was found:

FLAG: "powershell.exe" -EncodedCommand VwByAGkAdABIAC0ATwB1AHQAcAB1AHQAIAAiAFMAaQBtAHUAbABhAHQAZQBkACAAbwBiAGYAdQBzAGMAYQB0AGUAZAAGAGUAeABIAGMAdQB0AGkAbwBuACIA

```
DeviceProcessEvents
| where DeviceName == "acolyte756"
| where Timestamp >= datetime(2025-05-25T09:14:02.3941533Z)
| project Timestamp, DeviceName, FileName, FolderPath, ProcessCommandLine
| order by Timestamp asc
```

Timestamp	May 25, 2025 10:14:05 AM
DeviceName	acolyte756
FileName	powershell.exe
FolderPath	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
ProcessCommandLine	"powershell.exe" -EncodedCommand VwByAGkAdABIAC0ATwB1AHQAcAB1AHQAIAAiAFMAaQBtAHUAbABhAHQAZQBkACAAbw

Flag 6 – Evasion via Legacy Scripting

Objective:

Detect usage of outdated script configurations likely intended to bypass modern controls.

What to Hunt:

Look for uncommon scripting version of PowerShell or execution flags that reduce oversight.

Thought:

Modern defenses expect modern behavior. Watch for forced downgrades or legacy runtime calls.

.... It wasn't hard, it's literally just under the last flag.

FLAG: **"powershell.exe" -Version 2 -NoProfile -ExecutionPolicy Bypass -NoExit**

DeviceProcessEvents

| **where** DeviceName == "acolyte756"

| **where** Timestamp >= datetime(2025-05-25T09:14:02.3941533Z)

| **project** Timestamp, DeviceName, FileName, FolderPath, ProcessCommandLine

| **order by** Timestamp asc

>	May 25, 2025 10:...	acolyte756	powershell.exe	C:\Windows\System32\...	"powershell.exe" -EncodedCommand VwByAGkAdABlAC0ATwB1AHQAcaB1AHC
>	May 25, 2025 10:...	acolyte756	powershell.exe	C:\Windows\System32\...	"powershell.exe" -Version 2 -NoProfile -ExecutionPolicy Bypass -NoExit

Flag 7 – Remote Movement Discovery

Objective:

Reveal the intruder's next target beyond the initial breach point.

What to Hunt:

Trace outbound command patterns that reference hostnames unfamiliar to the local machine.

Thought:

Lateral movement often hides in plain sight. Connections to the right system at the wrong time can be the giveaway.

After pivoting from the previously established timestamp (2025-05-25T09:14:02.3941533Z), I investigated outbound connections using the direction field inside the *AdditionalFields* column in the *DeviceNetworkEvents*. This revealed a key outbound connection made to query the DNS of another Azure Virtual Machine, **victor-disa-vm.p2zfvso05mlezjev3ck4vqd3kd.cx.internal.cloudapp.net**.

FLAG: **victor-disa-vm**

DeviceNetworkEvents

| **where** DeviceName == "acolyte756"

| **where** Timestamp >= datetime(2025-05-25T09:29:43.3067155Z)

| **extend** NetType = parse_json(AdditionalFields)["direction"]

| **where** NetType == "Out"

| **order by** Timestamp asc

LocalPort	61517
Protocol	Udp
InitiatingProcessId	0
InitiatingProcessParentId	0
InitiatingProcessTokenEle...	None
ReportId	4974
✓ AdditionalFields	["direction":"Out","trans_id":"29921","query":"victor-disa-vm.p2zfvso05mlezjev3ck4vqd3kd.cx.internal.cloudapp.net","qclass":"1","qclass_name":"C_INTERNET","qtype":"1","
direction	Out
trans_id	29921
query	victor-disa-vm.p2zfvso05mlezjev3ck4vqd3kd.cx.internal.cloudapp.net
qclass	1
qclass_name	C_INTERNET
qtype	1
qtype_name	A

Flag 8 – Entry Indicators on Second Host

Objective:

Identify the subtle digital footprints left during a pivot.

What to Hunt:

Artifacts with naming patterns that imply staging, sync, or checkpointing.

Thought:

Every move leaves a mark — even if that mark is as simple as a filename that doesn't belong.

Hint:

1. point

After raging for 2 days and overcomplicating things (thanks a lot ChatGPT), I exported all the results from the KQL mentioned next into [Row Zero](#) (a better CSV visualizer, I usually use for exporting large data quantities from grafana and other data sources). And after filtering for point, I was surprised for seeing file: FLAG: **savepoint_sync.lnk**

```
DeviceFileEvents
```

```
| where DeviceName == "victor-disa-vm"
| where Timestamp >= datetime(2025-05-25T09:29:43.3067155Z)
| project Timestamp, FileName, FolderPath
| order by Timestamp asc
```


File

Edit

View

Data

Insert

Help

Undo

Redo

B

U

-

13

+

A

▼

▼

▼

▼

\$

%

?

0

00

Automatic

▼

↺

↻

↻

↻

T

+

↓

≡

🔍

📄

🔗

🔄

✕

📱

B2887

fx

savepoint_sync.Ink

	A	B	C
1	Timestamp	FileName	FolderPath
205	May 26, 2025 9:48:38 AM	JsonPointer.Net.dll	ppAppInstaller.1.25.390.0_x64_8wekyb3d8bbwe\Configur
338	May 26, 2025 9:48:38 AM	System.Net.ServicePoint.dll	ppAppInstaller.1.25.390.0_x64_8wekyb3d8bbwe\Configur
2294	May 26, 2025 2:19:34 PM	Microsoft.SharePoint.exe	rive\21.220.1024.0005\Microsoft.SharePoint.exe
2295	May 26, 2025 2:19:34 PM	Microsoft.SharePoint.exe	rive\21.220.1024.0005\Microsoft.SharePoint.exe
2296	May 26, 2025 2:19:34 PM	Microsoft.SharePoint.NativeMessagingClient.e	rive\21.220.1024.0005\Microsoft.SharePoint.NativeMessa
2297	May 26, 2025 2:19:34 PM	Microsoft.SharePoint.NativeMessagingClient.e	rive\21.220.1024.0005\Microsoft.SharePoint.NativeMessa
2499	May 26, 2025 2:18:25 PM	Microsoft.SharePoint.Calc.dll	rive\25.080.0427.0003\Microsoft.SharePoint.Calc.dll
2500	May 26, 2025 2:18:25 PM	Microsoft.SharePoint.dll	rive\25.080.0427.0003\Microsoft.SharePoint.dll
2501	May 26, 2025 2:18:25 PM	Microsoft.SharePoint.exe	rive\25.080.0427.0003\Microsoft.SharePoint.exe
2502	May 26, 2025 2:18:25 PM	Microsoft.SharePoint.HttpSvr.dll	rive\25.080.0427.0003\Microsoft.SharePoint.HttpSvr.dll
2503	May 26, 2025 2:18:25 PM	Microsoft.SharePoint.NativeMessagingClient.e	rive\25.080.0427.0003\Microsoft.SharePoint.NativeMessa
2504	May 26, 2025 2:18:25 PM	Microsoft.SharePoint.WebSocketClient.dll	rive\25.080.0427.0003\Microsoft.SharePoint.WebSocketC
2881	May 26, 2025 2:47:59 AM	checkpoint.lock.Ink	indows\Recent\checkpoint.lock.Ink
2887	May 26, 2025 2:47:51 AM	savepoint_sync.Ink	indows\Recent\savepoint_sync.Ink
3021	May 26, 2025 2:16:09 AM	CheckpointNotes_v43_578.csv	iativeTracking\2025-01\CheckpointNotes_v43_578.csv
3022	May 26, 2025 2:16:08 AM	CheckpointNotes_v4_628.txt	iativeTracking\2025-05\CheckpointNotes_v4_628.txt
3036	May 26, 2025 2:16:08 AM	CheckpointNotes_v12_462.csv	kManagement\2025-01\CheckpointNotes_v12_462.csv

Sort

Sort A-Z

Sort Z-A

Filter

+ Add condition

Search...

✕

☐ (Select all)

☒ checkpoint.lock.Ink

☒ CheckpointNotes_v12_462.csv

☒ CheckpointNotes_v4_628.txt

☒ CheckpointNotes_v43_578.csv

☒ JsonPointer.Net.dll

☒ Microsoft.SharePoint.Calc.dll

☒ Microsoft.SharePoint.dll

☒ Microsoft.SharePoint.exe

☒ Microsoft.SharePoint.HttpSvr.dll

Reset

Apply filter

Flag 8.1 – Persistence Registration on Entry

Objective:

Detect attempts to embed control mechanisms within system configuration.

What to Hunt:

Registry values tied to files or commands that were not present days before.

Thought:

Nothing says ownership like persistence. Look for traces that don't match the system's normal operational cadence.

Hint:

1. Utilize previous findings

It was actually really simple, I just searched for Registry Values that contained "savepoint".

FLAG: powershell.exe -NoProfile -ExecutionPolicy Bypass -File

"C:\Users\Public\savepoint_sync.ps1"

```
DeviceRegistryEvents
```

```
| where DeviceName == "victor-disa-vm"
| where Timestamp >= datetime(2025-05-25T09:14:02.3941533Z)
| where RegistryValueName contains "savepoint" or RegistryValueData contains "savepoint"
```

Flag 9 – External Communication Re-established

Objective:

Verify if outbound signals continued from the newly touched system.

What to Hunt:

Remote destinations not associated with the organization's known assets.

Thought:

When one door closes, another beacon opens. Follow the whispers outbound.

Hint:

1. Utilize previous findings

I searched for all the networked events on “victor-disa-vm” and then filter for all connections that the “RemoteUri” is not blank, after comparing to the previous machines, the domain was found:

FLAG: **eo1v1texlrdq3v.m.pipedream.net**

DeviceNetworkEvents

```
| where DeviceName == "victor-disa-vm"
| where Timestamp >= datetime(2025-05-25T09:14:02.3941533Z)
| where RemoteUri != ""
| order by Timestamp asc
```

Timestamp	DeviceId	DeviceName	ActionType	RemoteIP	RemotePort	RemoteUri	LocalIP
May 27, 2025 8:00...	7d385a602e4bec683...	victor-disa-vm	ConnectionFailed	23.96.180.189	443	fd.apiiris.microsoft.com	10.0.0.130
May 27, 2025 8:4...	7d385a602e4bec683...	victor-disa-vm	ConnectionFailed	20.96.153.111	443	fd.apiiris.microsoft.com	10.0.0.130
May 26, 2025 8:0...	7d385a602e4bec683...	victor-disa-vm	ConnectionSuccess	13.107.253.41	443	fb-unicast.msedge.net	10.0.0.130
May 26, 2025 3:2...	7d385a602e4bec683...	victor-disa-vm	ConnectionSuccess	3.215.219.189	443	eo1v1texlrdq3v.m.pipedream.net	10.0.0.130
May 26, 2025 3:4...	7d385a602e4bec683...	victor-disa-vm	ConnectionSuccess	23.20.75.208	443	eo1v1texlrdq3v.m.pipedream.net	10.0.0.130
May 26, 2025 3:4...	7d385a602e4bec683...	victor-disa-vm	ConnectionSuccess	54.165.125.87	443	eo1v1texlrdq3v.m.pipedream.net	10.0.0.130
May 26, 2025 4:4...	7d385a602e4bec683...	victor-disa-vm	ConnectionSuccess	54.89.6.16	443	eo1v1texlrdq3v.m.pipedream.net	10.0.0.130
May 26, 2025 4:4...	7d385a602e4bec683...	victor-disa-vm	ConnectionSuccess	54.165.125.87	443	eo1v1texlrdq3v.m.pipedream.net	10.0.0.130
May 26, 2025 5:4...	7d385a602e4bec683...	victor-disa-vm	ConnectionSuccess	174.129.85.136	443	eo1v1texlrdq3v.m.pipedream.net	10.0.0.130

Flag 10 – Stealth Mechanism Registration

Objective:

Uncover non-traditional persistence mechanisms leveraging system instrumentation.

What to Hunt:

Execution patterns or command traces that silently embed PowerShell scripts via background system monitors.

Thought:

Some persistence methods don't rely on scheduled tasks or run keys. Instead, they exploit Windows Management Instrumentation (WMI) to bind code to system behavior — event filters, consumers, and bindings quietly forming a re-execution trap.

If successful, the attacker no longer needs a login or shell to keep control.

Hint:

1. Report has it that a program containing a name "beacon" has done similar activity in other departments.

Because of the hint, i filtered all the process events that contained “beacon” on the ProcessCommandLine or InitiatingProcessCommandLine, the activities of "C:\Users\Public\beacon_sync_job_flag2.ps1" was found, and the earliest activity time was:

FLAG: **2025-05-26T02:48:07.2900744Z**

DeviceProcessEvents

```
| where DeviceName == "victor-disa-vm"
| where Timestamp >= datetime(2025-05-25T09:14:02.3941533Z)
| where ProcessCommandLine contains "beacon" or InitiatingProcessCommandLine contains "beacon"
| order by Timestamp asc
```

Getting started

Results

Query history

Export

Show empty columns

97 items

Search

00:03.570

Low

Chart type

Filters:

Add filter

nInfoProductV...	ProcessVersionInfoInternalFi...	ProcessVersionInfoOriginalFi...	ProcessVersionInfoFileDescri...	ProcessId	ProcessCommandLine
3996	POWERSHELL	PowerShell.EXE	Windows PowerShell	7996	"powershell.exe" -ExecutionPolicy Bypass -NoProfile -File "C:\Users\Public\beacon_sync.ps1"
3996	POWERSHELL	PowerShell.EXE	Windows PowerShell	8528	powershell.exe -WindowStyle Hidden -ExecutionPolicy Bypass -File "C:\Users\Public\beacon_sync_job_flag2.ps1"
3996	POWERSHELL	PowerShell.EXE	Windows PowerShell	9180	powershell.exe -WindowStyle Hidden -ExecutionPolicy Bypass -File "C:\Users\Public\beacon_sync_job_flag2.ps1"
3996	POWERSHELL	PowerShell.EXE	Windows PowerShell	2824	powershell.exe -WindowStyle Hidden -ExecutionPolicy Bypass -File "C:\Users\Public\beacon_sync_job_flag2.ps1"
3996	POWERSHELL	PowerShell.EXE	Windows PowerShell	8880	powershell.exe -WindowStyle Hidden -ExecutionPolicy Bypass -File "C:\Users\Public\beacon_sync_job_flag2.ps1"
3996	POWERSHELL	PowerShell.EXE	Windows PowerShell	8576	powershell.exe -WindowStyle Hidden -ExecutionPolicy Bypass -File "C:\Users\Public\beacon_sync_job_flag2.ps1"
3996	POWERSHELL	PowerShell.EXE	Windows PowerShell	2928	powershell.exe -WindowStyle Hidden -ExecutionPolicy Bypass -File "C:\Users\Public\beacon_sync_job_flag2.ps1"

Flag 11 – Suspicious Data Access Simulation

Objective:

Detect test-like access patterns mimicking sensitive credential theft.

What to Hunt:

References or interactions with files suggestive of password storage or system secrets.

Thought:

Even simulations create signals. Mimicry of real attacks is often part of preparation.

Hint:

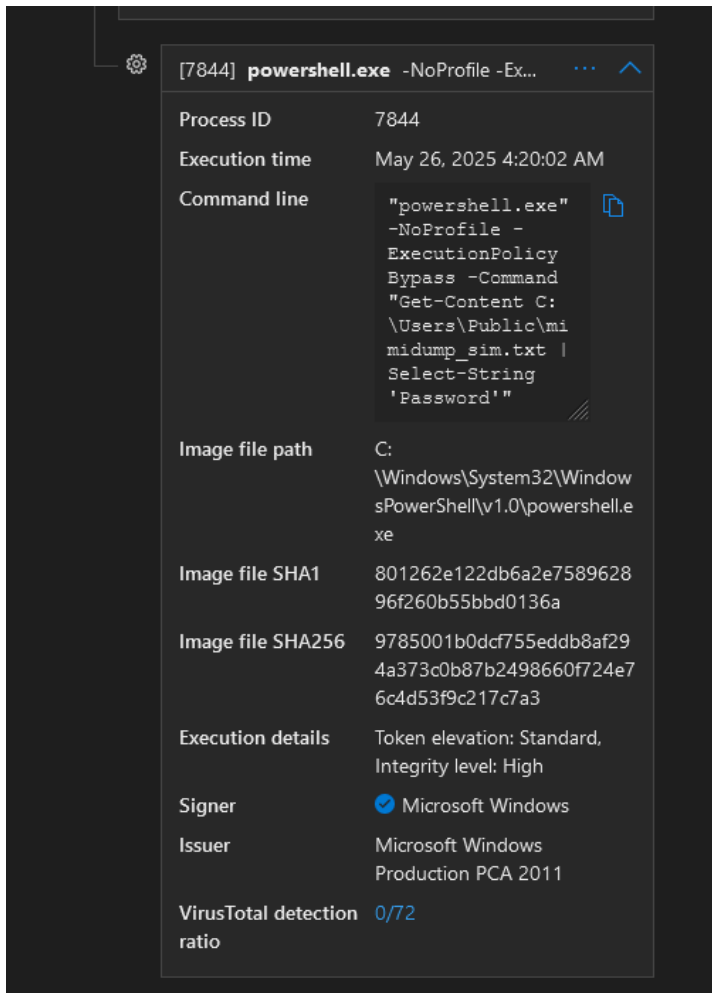
1. Possible Mimikatz variations

I searched for processes containing the initials 'mimi' on the FileName or ProcessCommandLine. Surprisingly, I was successful.

FLAG: **mimidump_sim.txt**

```
DeviceProcessEvents
```

```
| where DeviceName == "victor-disa-vm"
| where Timestamp >= datetime(2025-05-25T09:14:02.3941533Z)
| where FileName contains "mimi" or ProcessCommandLine contains "mimi"
| order by Timestamp asc
```



Flag 12 – Unusual Outbound Transfer

Objective:

Investigate signs of potential data transfer to untrusted locations.

What to Hunt:

External destinations indicative of third-party file storage or sharing services.

Thought:

The hands that take may hide in common destinations. Even familiar URLs can hide foul intent.

I mean... it's literally the SHA256 value of the event of the previous flag, so... I think further explanation is unnecessary.

FLAG: **9785001b0dcf755eddb8af294a373c0b87b2498660f724e76c4d53f9c217c7a3**

Flag 13 – Sensitive Asset Interaction

Objective:

Reveal whether any internal document of significance was involved.

What to Hunt:

Access logs involving time-sensitive or project-critical files.

Thought:

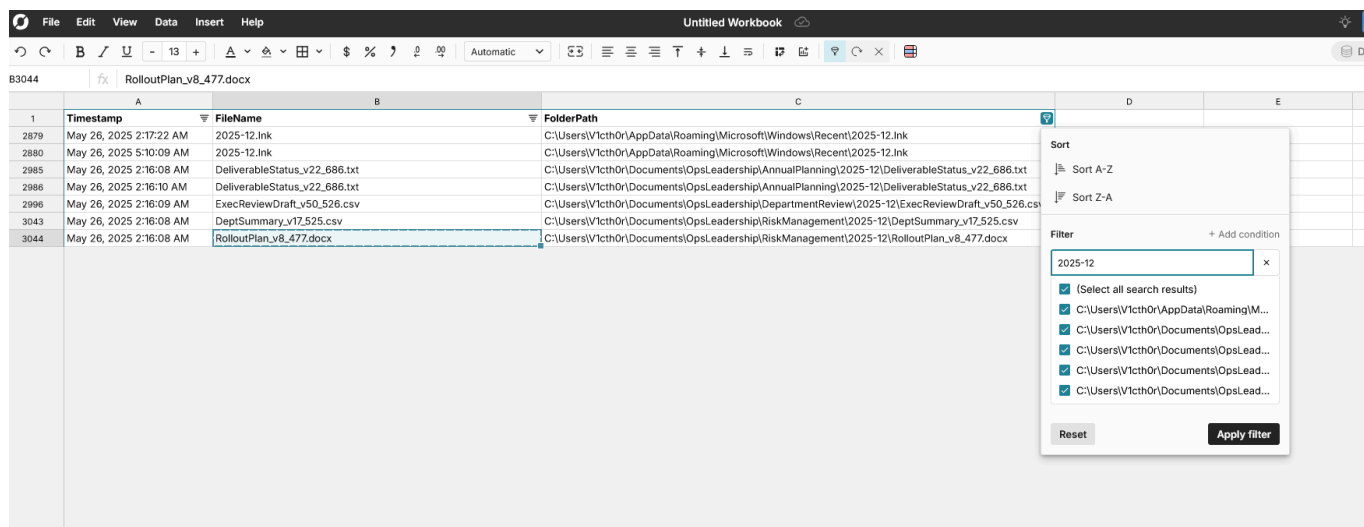
When the adversary browses project plans, it's not just about access — it's about intent.

Hint:

1. Organization suspects the adversary may have been targeting a document related to this year's

end month projects (yyyy-mm)

I still had the complete DeviceFileEvents table from Flag8 opened. After that I just searched for this year's end month (2025-12) in the folder filter and bingo! The file **RolloutPlan_v8_477.docx** was almost shining. FLAG: **RolloutPlan_v8_477.docx**



	A	B	C	D	E
1	Timestamp	FileName	FolderPath		
2879	May 26, 2025 2:17:22 AM	2025-12.Ink	C:\Users\V1cth0r\AppData\Roaming\Microsoft\Windows\Recent\2025-12.Ink		
2880	May 26, 2025 5:10:09 AM	2025-12.Ink	C:\Users\V1cth0r\AppData\Roaming\Microsoft\Windows\Recent\2025-12.Ink		
2985	May 26, 2025 2:16:08 AM	DeliverableStatus_v22_686.txt	C:\Users\V1cth0r\Documents\OpsLeadership\AnnualPlanning\2025-12\DeliverableStatus_v22_686.txt		
2986	May 26, 2025 2:16:10 AM	DeliverableStatus_v22_686.txt	C:\Users\V1cth0r\Documents\OpsLeadership\AnnualPlanning\2025-12\DeliverableStatus_v22_686.txt		
2996	May 26, 2025 2:16:09 AM	ExecReviewDraft_v50_526.csv	C:\Users\V1cth0r\Documents\OpsLeadership\DepartmentReview\2025-12\ExecReviewDraft_v50_526.csv		
3043	May 26, 2025 2:16:08 AM	DeptSummary_v17_525.csv	C:\Users\V1cth0r\Documents\OpsLeadership\RiskManagement\2025-12\DeptSummary_v17_525.csv		
3044	May 26, 2025 2:16:08 AM	RolloutPlan_v8_477.docx	C:\Users\V1cth0r\Documents\OpsLeadership\RiskManagement\2025-12\RolloutPlan_v8_477.docx		

Flag 14 – Tool Packaging Activity

Objective:

Spot behaviors related to preparing code or scripts for movement.

What to Hunt:

Compression or packaging of local assets in non-administrative directories.

Thought:

Before things move, they are prepared. Track the moment code becomes cargo.

I filtered for processes that had any mention of compression on the ProcessCommandLine and was successful.

FLAG: **"powershell.exe" -NoProfile -ExecutionPolicy Bypass -Command Compress-Archive -Path "C:\Users\Public\dropzone_spicy" -DestinationPath "C:\Users\Public\spicycore_loader_flag8.zip" -Force**

```
DeviceProcessEvents
| where DeviceName == "victor-disa-vm"
| where Timestamp >= datetime(2025-05-25T09:29:43.3067155Z)
| where ProcessCommandLine has_any ("7z", "zip", "tar")
| project Timestamp, InitiatingProcessAccountName, FileName, ProcessCommandLine,
FolderPath
| sort by Timestamp desc
```

Timestamp	InitiatingProcessAccountNa...	FileName	ProcessCommandLine
> May 26, 2025 5:4...	system	CollectGuestLogs.exe	"CollectGuestLogs.exe" -Mode:ga -FileName:D:\CollectGuestLogsTemp\VMAgentLogs.zip
> May 26, 2025 4:4...	system	CollectGuestLogs.exe	"CollectGuestLogs.exe" -Mode:ga -FileName:D:\CollectGuestLogsTemp\VMAgentLogs.zip
> May 26, 2025 1:4...	system	7z.exe	"7z.exe" a C:\ProgramData\employee-data-20250526124923.zip C:\ProgramData\employee-data-temp20250526124923.csv
> May 26, 2025 9:5...	system	7z.exe	"7z.exe" a C:\ProgramData\employee-data-20250526084957.zip C:\ProgramData\employee-data-temp20250526084957.csv
> May 26, 2025 7:0...	v1cth0r	powershell.exe	"powershell.exe" -NoProfile -ExecutionPolicy Bypass -Command Compress-Archive -Path "C:\Users\Public\dropzone_spicy" -DestinationPath "C:\Users\Public\spicycore_..."
> May 26, 2025 5:4...	system	7z.exe	"7z.exe" a C:\ProgramData\employee-data-20250526044944.zip C:\ProgramData\employee-data-temp20250526044944.csv
> May 26, 2025 1:4...	system	7z.exe	"7z.exe" a C:\ProgramData\employee-data-20250526004915.zip C:\ProgramData\employee-data-temp20250526004915.csv
> May 26, 2025 12:...	system	CollectGuestLogs.exe	"CollectGuestLogs.exe" -Mode:ga -FileName:D:\CollectGuestLogsTemp\VMAgentLogs.zip
> May 25, 2025 11:...	system	CollectGuestLogs.exe	"CollectGuestLogs.exe" -Mode:ga -FileName:D:\CollectGuestLogsTemp\VMAgentLogs.zip

Flag 15 – Deployment Artifact Planted

Objective:

Verify whether staged payloads were saved to disk.

What to Hunt:

Unusual file drops, especially compressed archives, in public or shared paths.

Thought:

Staged doesn't mean executed — yet. But it's the clearest sign something is about to happen.

Hint:

1. Utilize previous findings

Again, inside the previous FLAG, so I think context is not necessary.

FLAG: **spicycore_loader_flag8.zip**

7d385a602e4bec683...	victor-disa-vm	FileCreated	lqu4zbs0.dll	C:\Windows\Temp\lqu4zbs0\lqu4zbs0.dll
7d385a602e4bec683...	victor-disa-vm	FileCreated	spicycore_loader_flag8.zip	C:\Users\Public\spicycore_loader_flag8.zip
7d385a602e4bec683...	victor-disa-vm	FileCreated	WER8433.tmp.dmp	C:\ProgramData\Microsoft\Windows\WER\Te...
7d385a602e4bec683...	victor-disa-vm	FileCreated	Report.wer	C:\ProgramData\Microsoft\Windows\WER\Re...
7d385a602e4bec683...	victor-disa-vm	FileCreated	WERA0B4.tmp.dmp	C:\ProgramData\Microsoft\Windows\WER\Te...

Flag 16 – Persistence Trigger Finalized

Objective:

Identify automation set to invoke recently dropped content.

What to Hunt:

Scheduled execution entries tied to non-standard script names.

Thought:

If something's waiting to run, it's worth knowing when and what will pull the trigger.

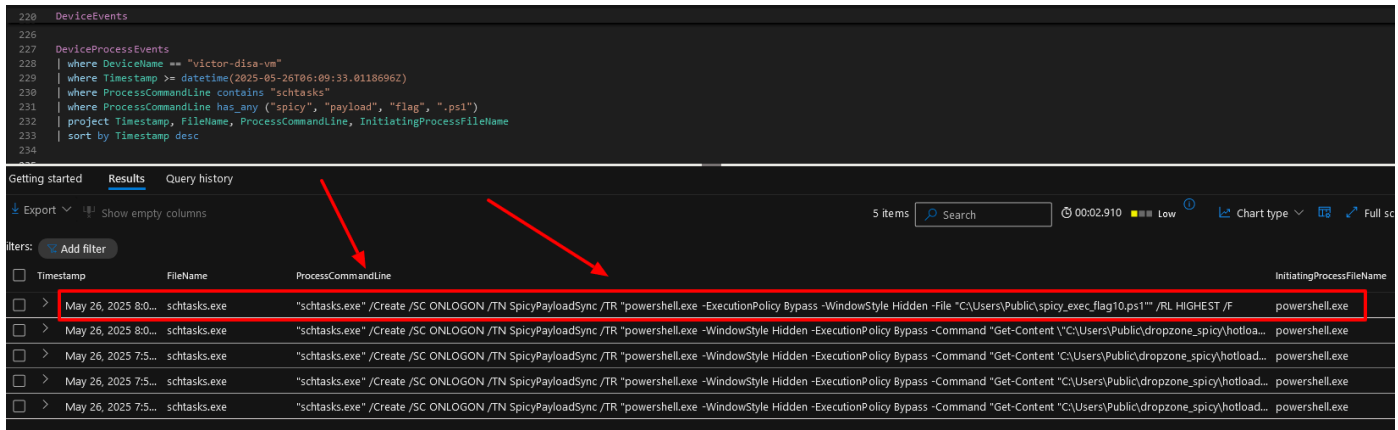
Hint:

1. Utilize previous findings

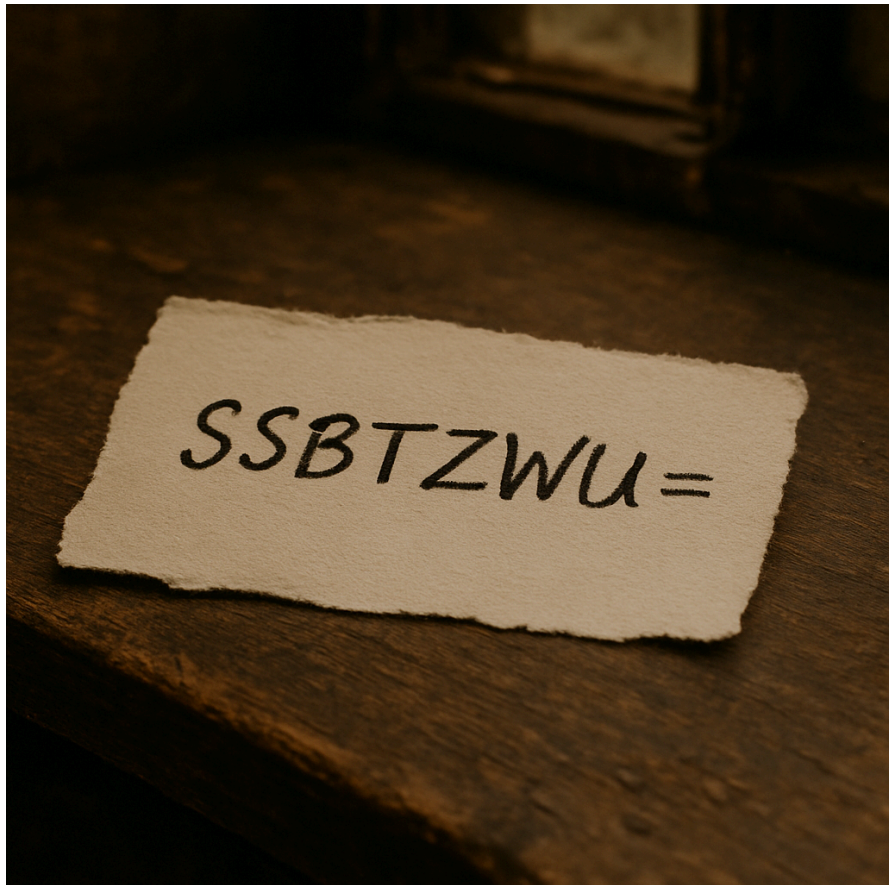
What I did to find it was filtering for Process Events that contained "schtasks" and then filter even further for .ps1, spicy and other "flag words".

```
DeviceProcessEvents
| where DeviceName == "victor-disa-vm"
| where Timestamp >= datetime(2025-05-26T06:09:33.0118696Z)
| where ProcessCommandLine contains "schtasks"
```

```
| where ProcessCommandLine has_any ("spicy", "payload", "flag", ".ps1")
| sort by Timestamp desc
```



“Secret Flag” Base64 - I SEE



TTP (Tactics, Techniques & Procedures)

Flag	Tactic	Technique Summary
1	Initial Access	Obfuscated PowerShell launched via Explorer, avoiding normal script execution.

2	Command and Control (C2)	Outbound HTTP beacon to external domain (pipedream.net).
3	Persistence	Registry run key set to auto-execute C2.ps1.
4	Persistence	Scheduled Task (SimC2Task) created for re-execution.
5	Defense Evasion	PowerShell -EncodedCommand used to hide script content.
6	Defense Evasion	Legacy PowerShell version (-Version 2) used to bypass logging and controls.
7	Lateral Movement	Internal DNS resolution to victor-disa-vm.
8	Discovery / Persistence	Dropped suspicious .lnk file (savepoint_sync.lnk) on new VM.
8.1	Persistence	Registry auto-exec entry pointing to savepoint_sync.ps1.
9	Command and Control (C2)	New outbound beacon from second host to pipedream.net.
10	Persistence	beacon_sync_job_flag2.ps1 run via WMI/Task triggers (likely via WMI binding).
11	Credential Access	Execution involving file named mimidump_sim.txt (Mimikatz-like simulation).
12	Exfiltration	Archive or file SHA256 indicates simulated transfer of credentials.
13	Collection	Access to sensitive document (RolloutPlan_v8_477.docx).
14	Exfiltration Preparation	Compression of local directory using Compress-Archive to .zip.
15	Staging	Artifact (spicycore_loader_flag8.zip) saved for movement.
16	Persistence	Scheduled task created to invoke .ps1 file as payload trigger.

IOC (Indicators of Compromise)

Type	Value
VM Name	acolyte756, victor-disa-vm

Registry Path	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\SimC2Task , registry keys containing savepoint
Scripts/Files	C2.ps1, savepoint_sync.ps1, beacon_sync_job_flag2.ps1, mimidump_sim.txt
Scheduled Task	SimC2Task
Archive Name	spicycore_loader_flag8.zip
Domains	eoqsu1hq6e9ulga.m.pipedream.net, eo1v1texxlrdq3v.m.pipedream.net
Document Accessed	RolloutPlan_v8_477.docx
PowerShell Flags	-EncodedCommand, -ExecutionPolicy Bypass, -Version 2

MITRE ATT&CK

Technique ID	Name	Flags
T1059.001	PowerShell	1, 5, 6
T1566	Phishing (or simulated delivery)	Entry assumption
T1105	Ingress Tool Transfer	2, 9, 12
T1027	Obfuscated Files or Information	5
T1070.004	Indicator Removal on Host	Implied by stealthy usage
T1547.001	Registry Run Keys / Startup Folder	3, 8.1
T1053.005	Scheduled Task/Job	4, 16
T1574.001	DLL Search Order Hijacking (implied)	Possibly via .lnk in Flag 8
T1218.013	Living off the Land: PowerShell	Throughout
T1021.001	Remote Services: SMB/Network Movement	7
T1555	Credential Dumping	11
T1005	Data from Local System	13
T1560.001	Archive Collected Data	14

T1074.001	Staging: Local Storage	15
T1546.003	Event Triggered Execution: WMI	10

Response Recommendations

Containment

- Isolate VMs **acolyte756** and **victor-disa-vm** from the network.
- Block outbound traffic to “*.pipedream.net” unless explicitly required and authorized by the company.

Forensics

- Collect memory dumps from both Virtual Machines.
- Dump Windows Event Logs, PowerShell logs, and Registry hives.
- Retrieve copies of suspicious files (*.ps1, .zip, .docx).

IOC Sweeping

- Hunt for any of the listed IOCs across the company (via Defender or Sentinel).
- Check for execution of **-EncodedCommand**, **-Version 2**, or **Compress-Archive** under suspicious accounts.

Persistence Audit

- Audit scheduled tasks and autorun registry entries.
- Enumerate WMI event subscriptions.

Remediation

- Remove malicious registry keys and tasks.
 - Delete suspicious artifacts from disk.
 - Reimage the hosts if tampering is confirmed.
-

Improvements & Lessons Learned

Detection Coverage

- Add alerts for:
 - PowerShell with **EncodedCommand**, **-Version 2**, or **Bypass**
 - Creation of Scheduled Tasks with **.ps1** or **Public** folder paths

- Any outbound DNS queries to dynamic domains (especially .pipedream.net and other similar)

Hunting SOP

- The use of a fixed baseline timestamp was extremely useful. Consider formalizing that as a baseline hunting technique in the company playbook.

Tooling

- **Row Zero** is a great hack for CSV-heavy workflows — consider integrating Grafana + Loki or similar for future scaled hunts.

Knowledge Sharing

- Turn this threat hunt into a blue team tabletop exercise or walkthrough training for others employees.